REPORT FOR CLOUD COMPUTING PROJECT

IAAS Cloud Middleware

Group A1, Team 2

Karan Ratnaparkhi
Supriya Kulkarni
Vedashree Govinda Gowda
Vibha Belavadi

Table of Contents

1 Introduction:

1.1 Aim of the project:

We are required to create an IaaS cloud middleware environment for job submission, scaling and migration using libvirt management API for providing resource management solutions. Also we are required to provide an interface for job submission, tracking the health of the VM, scaling and migration. Obtaining workloads, we should be able to perform cloud middleware management and analyze the results.

1.2 Milestones accomplished:

- Before Midterm Demo:

  A. Installed CentOS 7 on master and slave nodes
  B. Installed Ganglia for monitoring master and slave nodes with their VMs.
  C. Installed NFS (Network File System) for shared storage across migration.
  D. Set up KVM environment on all nodes for virtualization support
  E. Set up bridged network and storage across the master and slave nodes
  F. Developed scripts for VM creation, scaling, placement and migration.
  G. Extracted monitoring data from Ganglia's files and for using it to make resource management decisions

- After Midterm Demo:

  A. Streamlined placement algorithm to make it more robust and work for making decisions on where to place the VM for both creation and migration
  B. Added hosts status check algorithm to monitor health of the hosts in cluster which makes migration decisions to achieve load balancing.
  C. Integrated the scaling decision and code to allow for scaling of CPU and Memory in the VMs
  D. Removed the dependency on Ganglia for monitoring, instead wrote module for data collection scripts depending on the functionalities requiring it
  E. Completed script for successful VM Migration and Load balancing occurring simultaneously
  F. Successfully setup NFS node in collaboration with A2, and able to run algorithms with A2's workload
  G. Developed a user and client UI dashboard for creation, submission of jobs and where VM data and other functionalities like migration, scaling, can be seen
  H. Creation of host and VM graphs for its parameters – CPU, Memory, Disk to be displayed on the user interface using the rrd files generated from ganglia.

2 System Architecture Diagram:

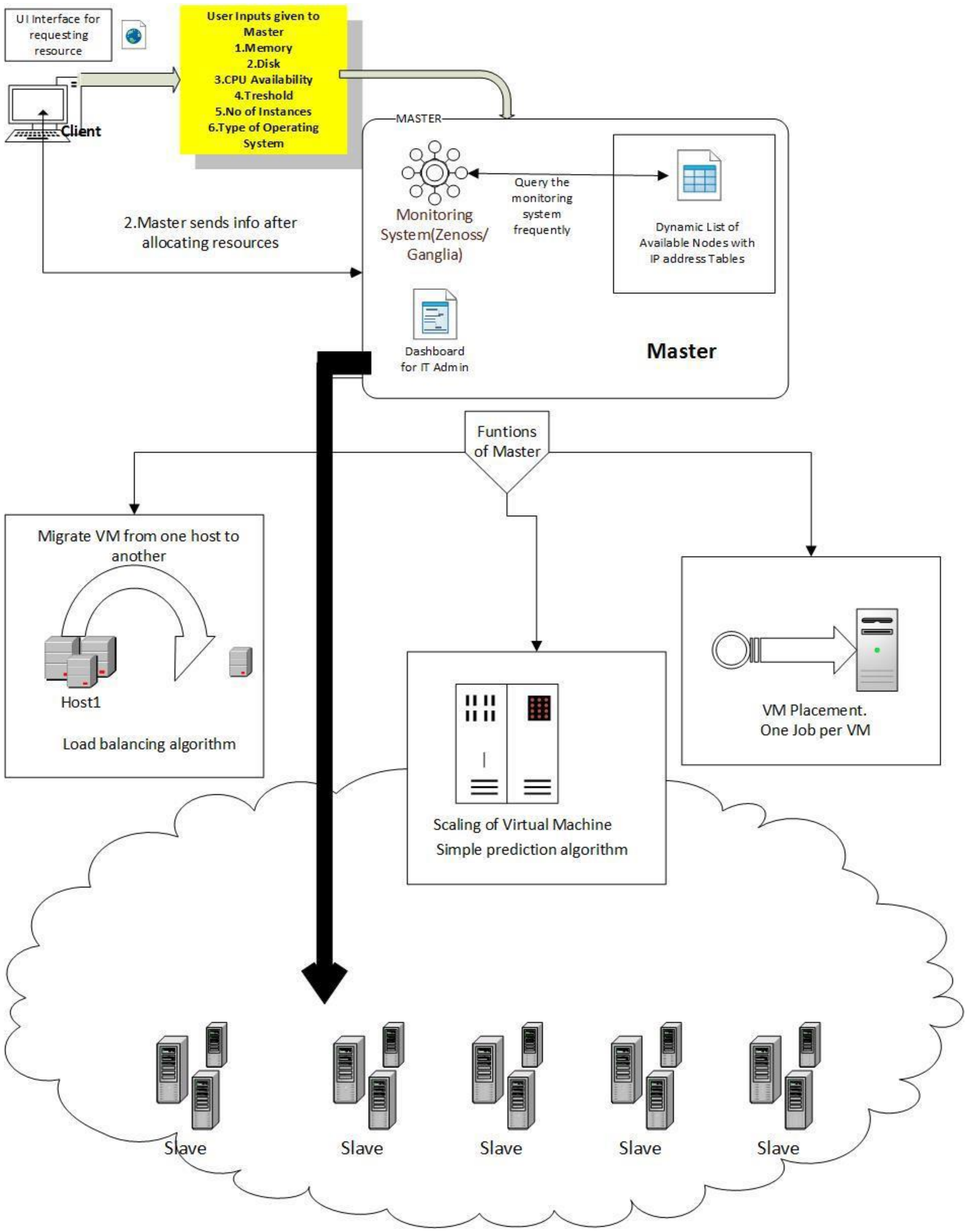


Figure 2.1 System Architecture and the components involved.

3 System Components:

[1] VM Creation:
[2] VM Placement
[3] VM Migration
[4] VM Scaling
[5] VM Monitoring
[6] VM Submission

Each of these functionalities will be explained in detail in the subsequent parts of the report.
To accomplish the project requirements, one of the host system is designated as the master node, with the remaining nodes being slaves to this master. The master node is responsible for the creation, scaling, migration and placement decisions of the virtual machines on the slave nodes as well as on itself. It uses the data collected by the monitoring system for all its resource management algorithms.

4 VM Creation and Placement:

The VM creation is the first and most preliminary step of the system, through which the user will be able to submit and run their jobs. The master node uses the user input information for VM creation decision. In addition to the above method, the user can also choose to provide an existing raw image of its VM through the NFS directory, which will then create VM that is hosted on the appropriate host. Creation of a VM can be achieved through libvirt API using the following command: **virsh create <XML config_file of VM>**

For creating a VM, the user requirements of the VM – VCPU, Memory, Disk, Virtual Machine name are taken as an input to the VM_create.py. If the client wishes to create/submit their VM, they can do so by submitting the raw image of their VM through the common directory that is shared between the client and the node system via NFS. Upon submitting the user requirements, the VM_Create.py triggers the placement.py script, which gives the appropriate host on which the VM. VM_create script is then called to create the VM. The user will be given the IP address of the newly created VM as decided dynamically by the DHCP system (from the script ConsoleForIP.exp) and the VM information file will be updated. In case the client, wants a copy of their VM, they are provided a range of IP addresses to choose from, and the VM created, will be assigned an IP Address in that range. VM creation script also involves calling the GnerateMac.py in order to generate a unique MAC ID for the VM.

Given below is the snapshot of the XML configuration file used for VM Creation by the master node:

node3-virt2.xml

```
domain type='kvm' id='11'>
  <name>node3-virt7</name>
  <uuid>VM_CREATE_UUID</uuid>
  <memory unit='KiB'>1048576</memory>
  <currentMemory unit='KiB'>1048576</currentMemory>
  <vcpu placement='static'>1</vcpu>
  <resource>
    <partition>/machine</partition>
  </resource>
  <os>
    <type arch='x86_64' machine='pc-i440fx-rhel7.0.0'>hvm</type>
    <boot dev='hd'/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <pae/>
  </features>
  <cpu mode='custom' match='exact'>
    <model fallback='allow'>SandyBridge</model>
  </cpu>
  <clock offset='utc'>
    <timer name='rtc' tickpolicy='catchup'/>
    <timer name='pit' tickpolicy='delay'/>
    <timer name='hpet' present='no'/>
  </clock>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>restart</on_crash>
  <pm>
    <suspend-to-mem enabled='no'/>
    <suspend-to-disk enabled='no'/>
  </pm>
  <devices>
    <emulator>/usr/libexec/qemu-kvm</emulator>
    <disk type='file' device='disk'>
      <driver name='qemu' type='qcow2'/>
      <source file='/home/node3/Downloads/CentOS-7-x86_64-GenericCloud-1503.qcow2'/>
      <backingStore/>
      <target dev='vda' bus='virtio'/>
      <alias name='virtio-disk0'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x07' function='0x0'/>
    </disk>
    <disk type='block' device='cdrom'>
```

5 VM Migration:

Migration is required for load balancing. The following diagram illustrates VM migration:

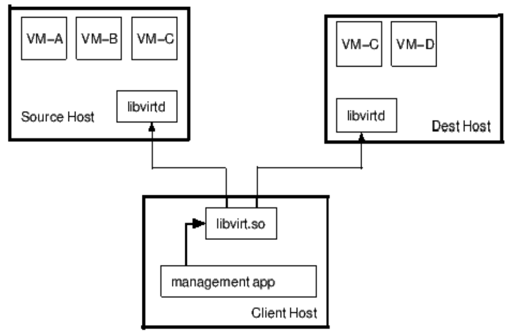


Figure 5.1 VM Migration between hosts

We use Pre-copy live migration as supported by libvirt in this project. This consists of two steps: Warm-up phase and the Stop-and-copy phase. In the Warm-up-phase, the source host copies the VM memory pages to the destination host while the VM is still running. The pages which get updated during this process (get ‘dirty’) are recopied. In the Stop-and-copy phase, the VM will be stopped on the original host and the remaining ‘dirty’ pages are copied to the destination.

Guest VM migration is possible only if the Virtual Machine’s storage (especially the one containing the virtual machine images) is shared between the hosts using NFS (Network File System) as the storage server. The KVM host server on both the hosts (source and dest) is configured and all the disk images of the guest are stored in the network shared folder. Thus in a way, these guest images will be accessed from source host even after migration.

Migration can be achieved through libvirt API using the virsh command:

ssh -q -o StrictHostKeyChecking=no root@source_host "virsh migrate vmid qemu+ssh://dest_host/system”

In this project, we have a HostPatrolling.py script that runs in the background and checks the host utilization to see if there is any need for migration. If the need arises, it calls the VM_migrate.py to get the VM to be migrated. VM_migrate.py in turn call the VM_placement.py to get the target host where the VM is supposed to migrate. Upon getting the target host IP Address, we then migrate the VM using the migrate_VM.sh. Migration, runs in conjunction with Scaling. This is to ensure that a VM which can be scaled within the host will do so and we avoid unnecessary cost in migrating VM to another host.

6 VM Scaling:

It is not always possible to exactly identify the CPU and RAM requirements at the time of deploying a VM. Dynamic scaling for CPU and RAM feature would allow us to change these resources for a running VM which will avoid any downtime. If scaling up requires a migration when the VM reaches the threshold for both memory and vcpu - if the current host where VM is running has capacity then simply the resources will be scaled up, if not live migration will be done within the cluster and then resources will be scaled up.

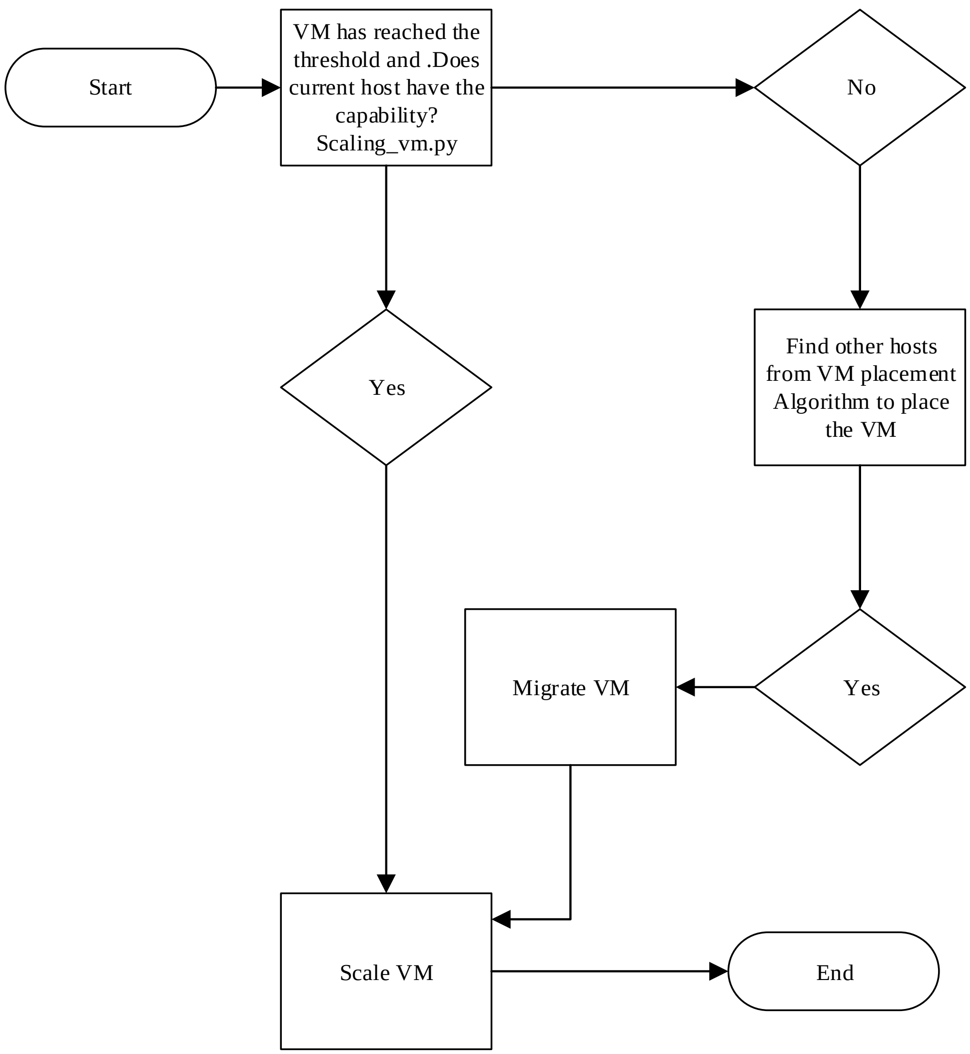


Figure 6.1 Scaling of virtual machines

7 VM Monitoring and data collection:

Initially, Ganglia was responsible for VM Monitoring. Ganglia has three main components: Gmond, Gmetad and ganglia-web frontend. Gmetad is the collector demon on the master node that collects all. Gmond is the monitoring daemon that sends all the monitoring data from the hosts and virtual machine. Ganglia-web-interface presents the data in a graphical format. However, there were some major disadvantages of using Ganglia. One of them was the 0-5 min latency in data collection. The other disadvantage was Ganglia crashing very frequently, and thus rendering data collection method obsolete.

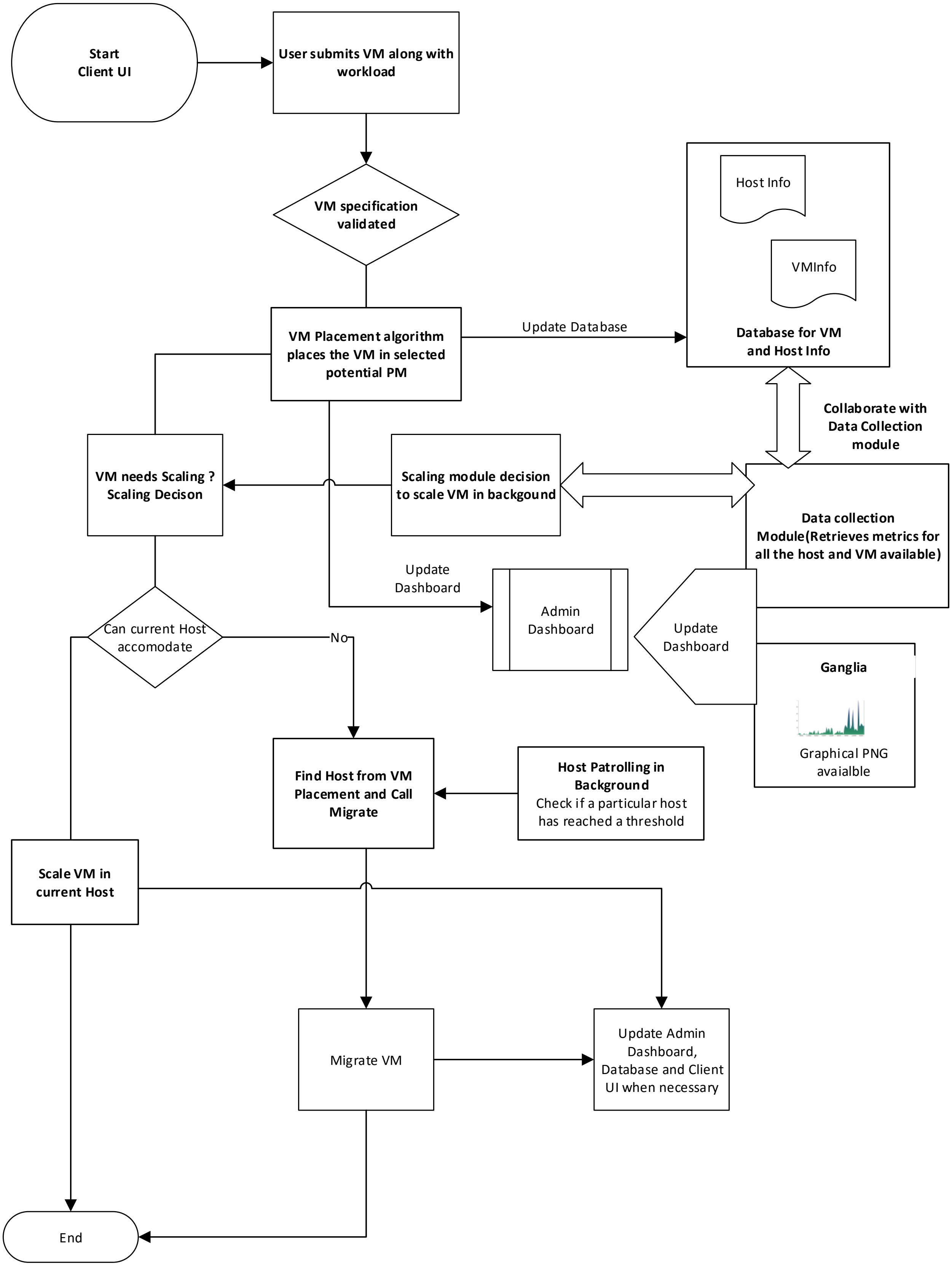
Due to these difficulties faced, we developed a suite of data collection scripts that provided the data according to the functionalities of VM creation, Migration, Scaling. Some of the most salient advantages include decoupling the functionality requirements and having a data collection method that is robust to node failure and overlapping network usage. We use system commands like free –m, top –bn 2, and df –h –total to get memory, CPU and disk utilization respectively. The only issue with this approach was since ssh to each

Ganglia is still being used in order to get RRDs of CPU, IO, Memory data of the hosts and the VMs for graph creation and display in the user interface. We use the rrdtool function to generate the graphs, that being the only dependency on Ganglia.

8 VM Submission:

A2 team members can submit their VM on this environment and can run the workloads. This is facilitated using NFS setup between cluster of A1 team and A2 team. Once we have access to image and XML file of the VM which needs to be submitted, we follow the VM creation procedure to start it on our system.

9 FLOWCHART OF SYSTEM



10 User Interface:

Admin Dashboard – User Interface for the admin for viewing all information and metrics related to VM and hosts in the Infrastructure Layer. The UI provides a graphical view of the VM metrics for the admin and also provides the capability to scale the VM.

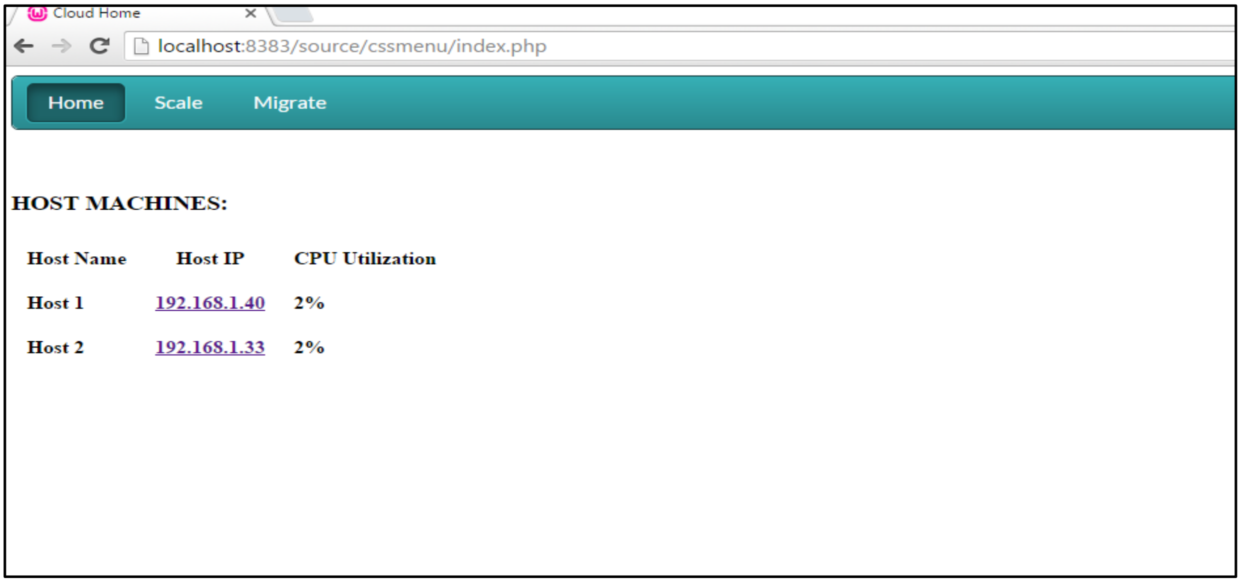


Figure 9.1 List of Host Machines in Infrastructure Layer

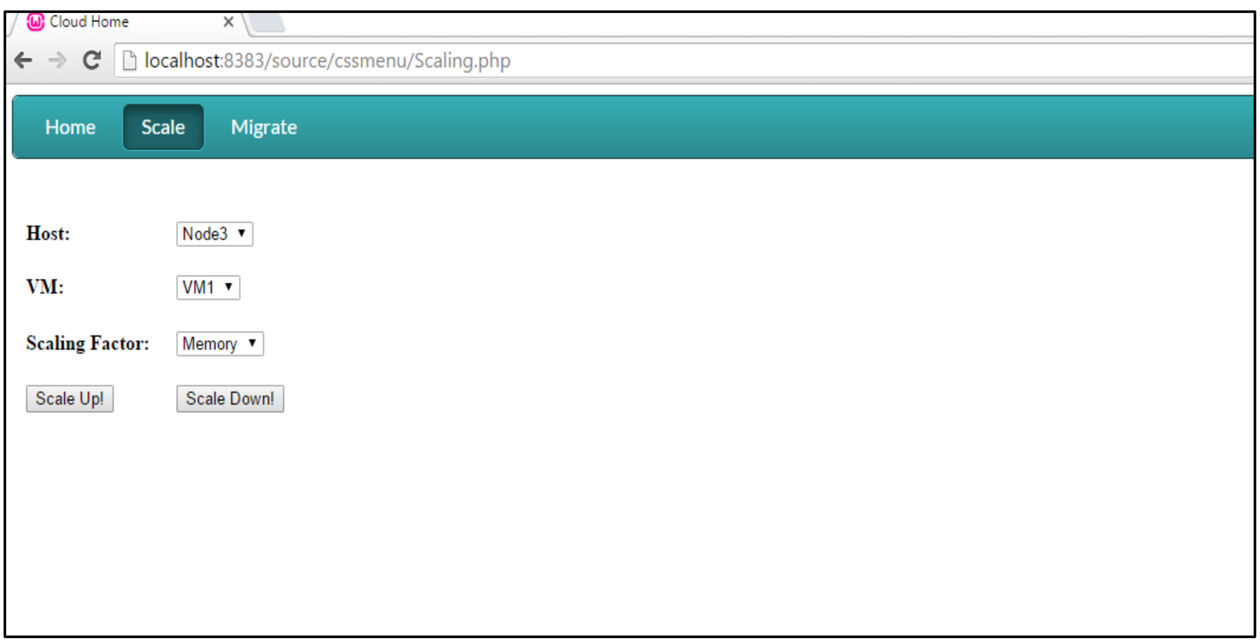


Figure 9.2 Admin Interface to scale a VM on any host

Figure 9.3: Admin Interface to migrate a VM from one host to another host

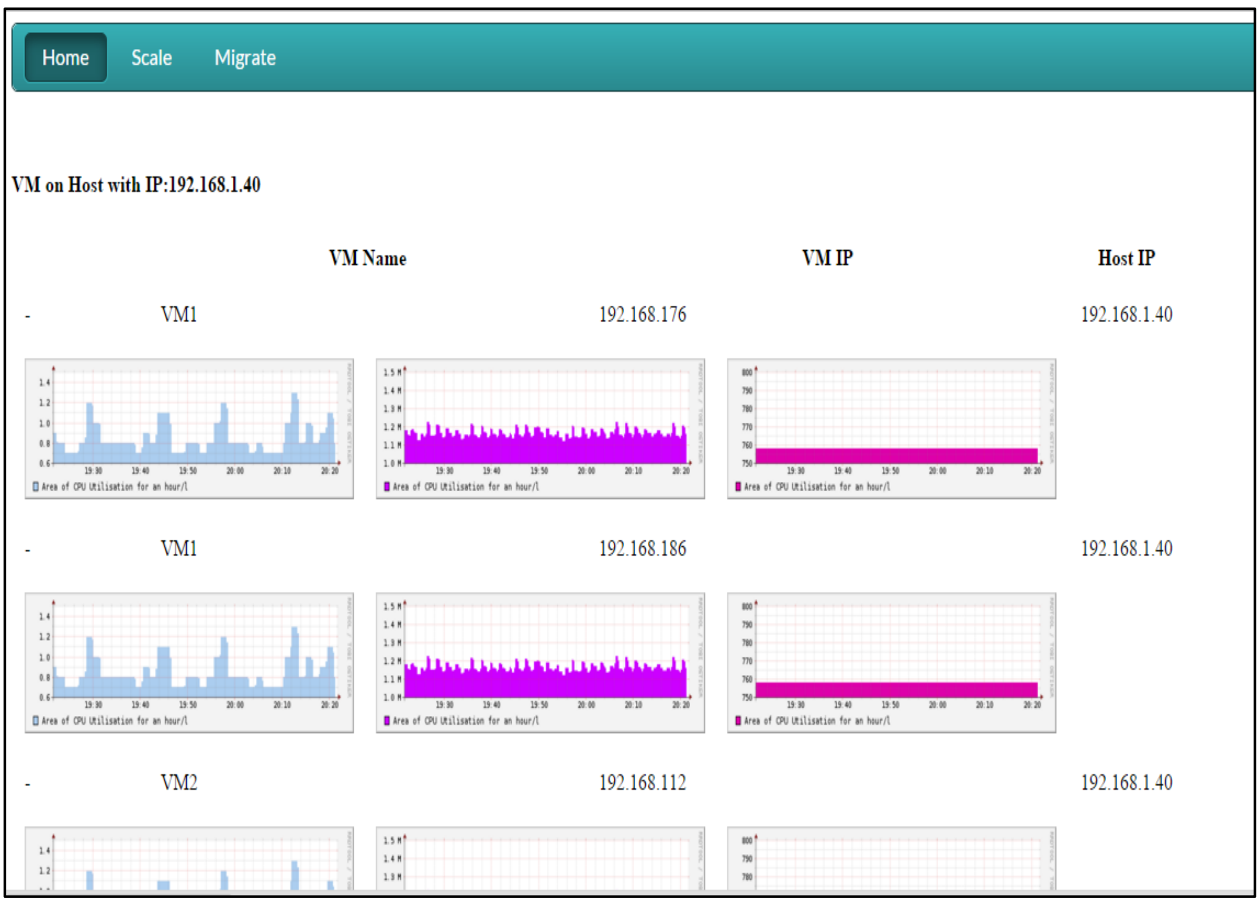


Figure 9.2 Graphical view of various metrics for VM in a particular host.

Client UI – User Interface for client to submit VM

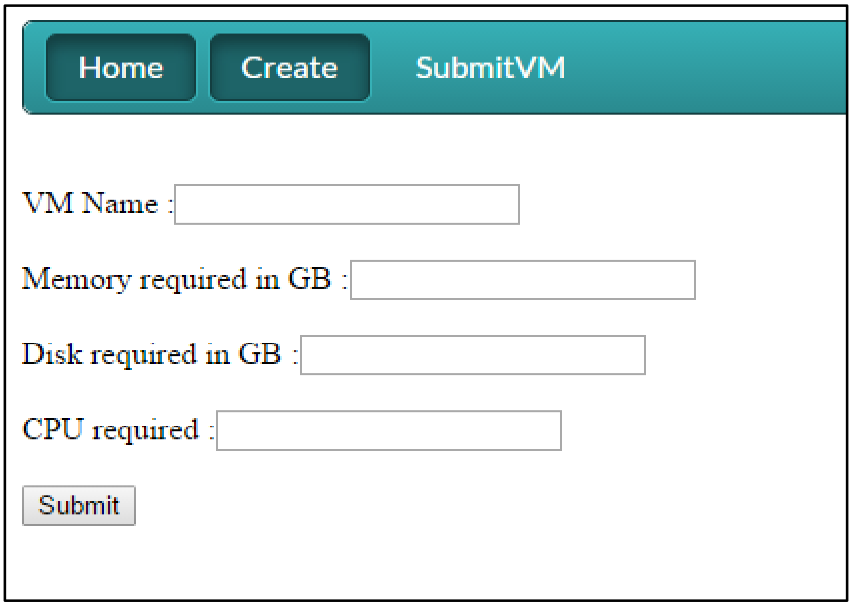


Figure9.4 Client UI to submit VM along with specifications

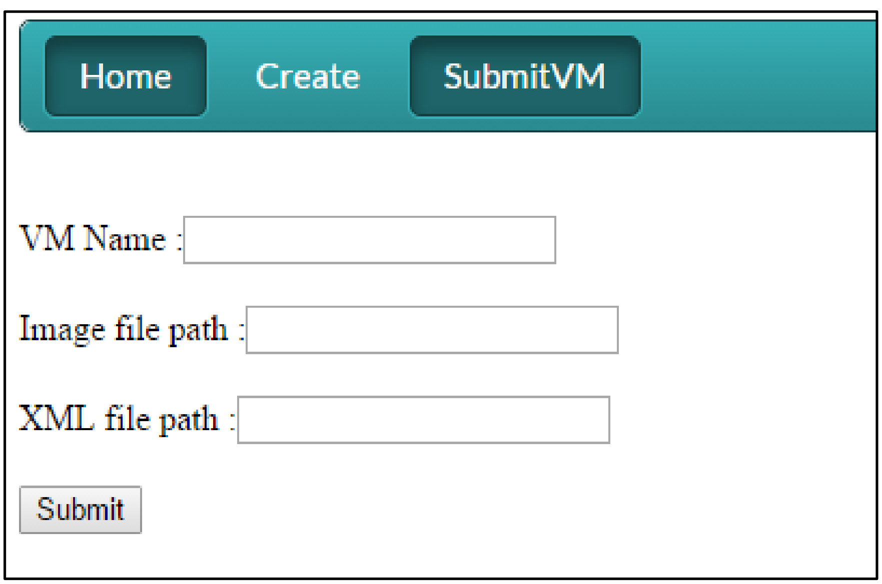


Figure 9.5 Client UI to submit workload

11 Setup information:

For doing the setup of a node – adding it into the cluster, we need to do the following:

[1] Updating the VMinfo.txt file to include the VMinfo
[2] Installing Ganglia
[3] Making passwordless ssh from the master to the VMs
[4] Configuring NFS on the client side system to support

12 Challenges Faced:

Problems with deliverables	Details on how it was fixed
Installation	Setting up bridge network, configuring static IP and port-forwarding.
Monitoring System	High CPU Utilization when Zenoss is used for monitoring hence went ahead with Ganglia.
VM Migration	Setting up NFS mount had the problem of RPC port mapper failure issue. Resolved by adding accept rules for firewall.
Ganglia Installation	Ganglia packages not available in CentOS7 repository. Installed epel-repository for the same.
Ganglia Setup	Faced issues during installation of Ganglia with configuring gmond. Resolved it by specifying the host IP address and commenting out the mcast option.
Ganglia port problem	Ganglia's listening port in conflict with XAMPP's port, hence went ahead and
Data Collection scripts stuck	Data collection program stuck as some of the VMs become dead after a while. Solved it by first getting an acknowledgment from the VM that it is alive and proceeding with the data collection.
ssh to each node for data collection taking a lot of time	Did passwordless ssh to all the nodes. Check IP address could be pinged Resolve ssh globally issue in by specifying DNS in the sshd_config file

13 Impacted Files:

- VM Creation and Placement:

Name.	Description
VM_create.py	Calls the placement algorithm to create a VM on the appropriate host
Createvm.sh	Creates the VM on the specified host using CPU, Memory, Disk parameters provided by the caller process.
baseImg.qcow2	Base image for VM creation
VM_Placement.py	Outputs a suitable host for a given Virtual Machine, used during submission and migration
Host_Info.txt	Stores host information including host IP address, Host name.
VM_Info.txt .	Stores virtual machine information including VM IP address, VM name, VM's host IP address.
ConsoleForIP.exp	Retrieves IP from newly created VM using virsh console functionality
GenerateMac.py	Generates a unique mac Id required for the VM
Get_ip.py	Extract free IP available in network

- VM Migration:

Name.	Description
VM_migrate.py	Initiates Migration of a virtual machine
HostPatroling.py	Checks the status of host machines and take the appropriate actions based on the host utilization of CPU and Memory which helps to load balance them
migratevm.sh	Migrates the VM on the host specified

- VM Scaling:

Name.	Description
DefineVM.sh	Bash script to define VM using the config XML. This script is used to scale down the VM
scaling_utilities.py	Contains utility functions for scaling of memory and CPU
scaling_vm.py	Decides whether or not to scale in current host or if migrate module needs to be called.
check_scalability.py	Checks if the vCPU of the current virtual machine can be scaled
parse_vcpu.py	Checks if vCPU can be scaled for the virtual machine
scaling_mem.py	Checks if scaling of memory is possible for the virtual machine
scale_machine.py	Scales memory of the virtual machine
scale_up_vm.sh	Scales vCPU of the virtual machine

- VM monitoring and data files:

Name.	Description
gangInst.sh	Installs ganglia and gmond on Virtual Machine, makes passwordless ssh from host to VM
GangGraphs.sh	Creates graphs from the rrd files stored by ganglia
RestartGan.sh	Routinely restarts ganglia gmetad and gmond on all the machines
VMKaran.sh	Gets the VM’s CPU utilization and memory utilization information for migration
ioscript.sh	Gives the CPU, disk and memory utilization of the host machines
VMDominfo.sh	Gives the vCPU and memory assignment for a Virtual Machine made during it’s creation
AvailMem.sh	Gives the available memory for the host machine
VMVeda.sh	Gets the maximum and current memory assignment to a virtual machine, and the

	current memory and CPU utilization of the Virtual Machine

14 References:

[1] On Theory of VM Placement: Anomalies in Existing Methodologies and Their Mitigation Using a Novel Vector Based Approach Appendix I: Process Details
[2] On Resource Management for Cloud Users: A Generalized Kelly Mechanism Approach
[3] SLA-aware virtual resource management for cloud infrastructures
[4] VMware Distributed Resource Management: Design, Implementation, and Lessons Learned
[5] Q-Clouds: Managing Performance Interference Effects for QoS-Aware Clouds
[6] https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/5/html/Virtualization/chap-Virtualization-Managing_guests_with_virsh.html
[7] http://libvirt.org/
[8] http://techblog.netflix.com/2012/01/auto-scaling-in-amazon-cloud.html

15 Individual Contributions:

Name	Work
Karan Ratnaparkhi	1.KVM/Cent-OS setup 2. Port forwarding 3.VM creation 4.NFS setup and troubleshooting 5.Load balancing of hosts (host patrolling) 6.Client UI 7.VM migration setup
Supriya Kulkarni	1. Installation of Operating System Centos and KVM 2. Setup Bridge network 3. VM Placement algorithm. 4. Maintain Database for VM and Host Information 5. Creation of Admin UI 6. Processing user Input and initiating creation and migration algorithm

Vedashree Govinda Gowda	1. Installation of Operating System, KVM 2. Configuring Bridge network 3. Comparison of Ganglia and Zenoss 4. Extraction of available IP address during VM Creation 5. Scaling of Memory, CPU of Virtual machines on all hosts. 6. Scaling Decision algorithm for load balancing
Vibha Belavadi	1. Installation of KVM and Operating System Centos 2. Setup of network on all hosts 3. Installation of monitoring system Ganglia and Zenoss 4. Comparison of Ganglia and Zenoss 5. Data Collection of Ganglia by parsing xml 6. Isolation of Ganglia form the data collection module. 7. Independent Data Collection module which collaborates and continuously provides metrics to other important decision algorithms 8. Module for generating graphs for required metrics from RRD files of Ganglia.